



(11) **EP 2 513 747 B1**

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention
of the grant of the patent:
21.02.2018 Bulletin 2018/08

(21) Application number: **10795853.0**

(22) Date of filing: **08.12.2010**

(51) Int Cl.:
G06F 1/32 (2006.01)

(86) International application number:
PCT/US2010/059562

(87) International publication number:
WO 2011/084336 (14.07.2011 Gazette 2011/28)

(54) **SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER WITH REDUCED FREQUENCY OSCILLATIONS**

SYSTEM UND VERFAHREN ZUR STEUERUNG DER LEISTUNG EINER ZENTRALEN
PROZESSOREINHEIT MIT REDUZIERTEN FREQUENZSCHWANKUNGEN

SYSTÈME ET PROCÉDÉ PERMETTANT DE COMMANDER LA PUISSANCE D'UNE UNITÉ
CENTRALE DE TRAITEMENT AVEC DES OSCILLATIONS DE FRÉQUENCE RÉDUITES

(84) Designated Contracting States:
**AL AT BE BG CH CY CZ DE DK EE ES FI FR GB
GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO
PL PT RO RS SE SI SK SM TR**

(30) Priority: **11.11.2010 US 944378**
16.12.2009 US 286979 P

(43) Date of publication of application:
24.10.2012 Bulletin 2012/43

(73) Proprietor: **Qualcomm Incorporated**
San Diego, CA 92121-1714 (US)

(72) Inventors:
• **THOMSON, Steven S.**
San Diego, California 92121-1714 (US)

- **RYCHLIK, Bohuslav**
San Diego, California 92121-1714 (US)
- **IRANLI, Ali**
San Diego, California 92121-1714 (US)
- **SALSBERY, Brian J.**
San Diego, California 92121-1714 (US)
- **SUR, Sumit**
San Diego, California 92121-1714 (US)
- **GARGASH, Norman S.**
San Diego, California 92121-1714 (US)

(74) Representative: **Tomkins & Co**
5 Dartmouth Road
Dublin 6 (IE)

(56) References cited:
EP-A2- 0 098 169 US-A1- 2006 123 253
US-A1- 2009 249 347

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

DescriptionDESCRIPTION OF THE RELATED ART

[0001] Portable computing devices (PDs) are ubiquitous. These devices may include cellular telephones, portable digital assistants (PDAs), portable game consoles, palmtop computers, and other portable electronic devices. In addition to the primary function of these devices, many include peripheral functions. For example, a cellular telephone may include the primary function of making cellular telephone calls and the peripheral functions of a still camera, a video camera, global positioning system (GPS) navigation, web browsing, sending and receiving emails, sending and receiving text messages, push-to-talk capabilities, etc. As the functionality of such a device increases, the computing or processing power required to support such functionality also increases. Further, as the computing power increases, there exists a greater need to effectively manage the processor, or processors, that provide the computing power.

[0002] Accordingly, what is needed is an improved method of controlling power within a CPU. An method of saving power according to the prior art is disclosed by the application US 2006/0123253.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] In the figures, like reference numerals refer to like parts throughout the various views unless otherwise indicated.

FIG. 1 is a front plan view of a first aspect of a portable computing device (PCD) in a closed position;

FIG. 2 is a front plan view of the first aspect of a PCD in an open position;

FIG. 3 is a block diagram of a second aspect of a PCD;

FIG. 4 is a block diagram of a processing system;

FIG. 5 is a flowchart illustrating a first aspect of a method of dynamically controlling power within a CPU;

FIG. 6 is a flowchart illustrating a first portion of a second aspect of a method of dynamically controlling power within a CPU;

FIG. 7 is a flowchart illustrating a second portion of the second aspect of a method of dynamically controlling power within a CPU;

FIG. 8 is a flowchart illustrating a third portion of the second aspect of a method of dynamically controlling power within a CPU; and

FIG. 9 is an exemplary graph showing the dynamic clock and voltage scaling (DCVS) controlled CPU frequency plotted over time.

DETAILED DESCRIPTION

[0004] The word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any aspect described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other aspects.

[0005] In this description, the term "application" may also include files having executable content, such as: object code, scripts, byte code, markup language files, and patches. In addition, an "application" referred to herein, may also include files that are not executable in nature, such as documents that may need to be opened or other data files that need to be accessed.

[0006] The term "content" may also include files having executable content, such as: object code, scripts, byte code, markup language files, and patches. In addition, "content" referred to herein, may also include files that are not executable in nature, such as documents that may need to be opened or other data files that need to be accessed.

[0007] As used in this description, the terms "component," "database," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device may be a component. One or more components may reside within a process and/or thread of execution, and a component may be localized on one computer and/or distributed between two or more computers. In addition, these components may execute from various computer readable media having various data structures stored thereon. The components may communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

[0008] Referring initially to FIG. 1 and FIG. 2, an exemplary portable computing device (PCD) is shown and is generally designated 100. As shown, the PCD 100 may include a housing 102. The housing 102 may include an upper housing portion 104 and a lower housing portion 106. FIG. 1 shows that the upper housing portion 104 may include a display

108. In a particular aspect, the display 108 may be a touch screen display. The upper housing portion 104 may also include a trackball input device 110. Further, as shown in FIG. 1, the upper housing portion 104 may include a power on button 112 and a power off button 114. As shown in FIG. 1, the upper housing portion 104 of the PCD 100 may include a plurality of indicator lights 116 and a speaker 118. Each indicator light 116 may be a light emitting diode (LED).

5 **[0009]** In a particular aspect, as depicted in FIG. 2, the upper housing portion 104 is movable relative to the lower housing portion 106. Specifically, the upper housing portion 104 may be slidable relative to the lower housing portion 106. As shown in FIG. 2, the lower housing portion 106 may include a multi-button keyboard 120. In a particular aspect, the multi-button keyboard 120 may be a standard QWERTY keyboard. The multi-button keyboard 120 may be revealed when the upper housing portion 104 is moved relative to the lower housing portion 106. FIG. 2 further illustrates that the

10 **[0010]** Referring to FIG. 3, an exemplary, non-limiting aspect of a portable computing device (PCD) is shown and is generally designated 320. As shown, the PCD 320 includes an on-chip system 322 that includes a multicore CPU 324. The multicore CPU 324 may include a zeroth core 325, a first core 326, and an Nth core 327.

15 **[0011]** As illustrated in FIG. 3, a display controller 328 and a touch screen controller 330 are coupled to the multicore CPU 324. In turn, a touch screen display 332 external to the on-chip system 322 is coupled to the display controller 328 and the touch screen controller 330.

20 **[0012]** FIG. 3 further indicates that a video encoder 334, e.g., a phase alternating line (PAL) encoder, a sequential couleur a memoire (SECAM) encoder, or a national television system(s) committee (NTSC) encoder, is coupled to the multicore CPU 324. Further, a video amplifier 336 is coupled to the video encoder 334 and the touch screen display 332. Also, a video port 338 is coupled to the video amplifier 336. As depicted in FIG. 3, a universal serial bus (USB) controller 340 is coupled to the multicore CPU 324. Also, a USB port 342 is coupled to the USB controller 340. A memory 344 and a subscriber identity module (SIM) card 346 may also be coupled to the multicore CPU 324. Further, as shown in FIG. 3, a digital camera 348 may be coupled to the multicore CPU 324. In an exemplary aspect, the digital camera 348 is a charge-coupled device (CCD) camera or a complementary metal-oxide semiconductor (CMOS) camera.

25 **[0013]** As further illustrated in FIG. 3, a stereo audio CODEC 350 may be coupled to the multicore CPU 324. Moreover, an audio amplifier 352 may be coupled to the stereo audio CODEC 350. In an exemplary aspect, a first stereo speaker 354 and a second stereo speaker 356 are coupled to the audio amplifier 352. FIG. 3 shows that a microphone amplifier 358 may be also coupled to the stereo audio CODEC 350. Additionally, a microphone 360 may be coupled to the microphone amplifier 358. In a particular aspect, a frequency modulation (FM) radio tuner 362 may be coupled to the stereo audio CODEC 350. Also, an FM antenna 364 is coupled to the FM radio tuner 362. Further, stereo headphones 366 may be coupled to the stereo audio CODEC 350.

30 **[0014]** FIG. 3 further indicates that a radio frequency (RF) transceiver 368 may be coupled to the multicore CPU 324. An RF switch 370 may be coupled to the RF transceiver 368 and an RF antenna 372. As shown in FIG. 3, a keypad 374 may be coupled to the multicore CPU 324. Also, a mono headset with a microphone 376 may be coupled to the multicore CPU 324. Further, a vibrator device 378 may be coupled to the multicore CPU 324. FIG. 3 also shows that a power supply 380 may be coupled to the on-chip system 322. In a particular aspect, the power supply 380 is a direct current (DC) power supply that provides power to the various components of the PCD 320 that require power. Further, in a particular aspect, the power supply is a rechargeable DC battery or a DC power supply that is derived from an alternating current (AC) to DC transformer that is connected to an AC power source.

35 **[0015]** FIG. 3 further indicates that the PCD 320 may also include a network card 388 that may be used to access a data network, e.g., a local area network, a personal area network, or any other network. The network card 388 may be a Bluetooth network card, a WiFi network card, a personal area network (PAN) card, a personal area network ultra-low-power technology (PeANUT) network card, or any other network card well known in the art. Further, the network card 388 may be incorporated into a chip, i.e., the network card 388 may be a full solution in a chip, and may not be a separate network card 388.

40 **[0016]** As depicted in FIG. 3, the touch screen display 332, the video port 338, the USB port 342, the camera 348, the first stereo speaker 354, the second stereo speaker 356, the microphone 360, the FM antenna 364, the stereo headphones 366, the RF switch 370, the RF antenna 372, the keypad 374, the mono headset 376, the vibrator 378, and the power supply 380 are external to the on-chip system 322.

45 **[0017]** In a particular aspect, one or more of the method steps described herein may be stored in the memory 344 as computer program instructions. These instructions may be executed by the multicore CPU 324 in order to perform the methods described herein. Further, the multicore CPU 324, the memory 344, or a combination thereof may serve as a means for executing one or more of the method steps described herein in order to dynamically control the power within a CPU, or core, of the multicore CPU 324.

50 **[0018]** Referring to FIG. 4, a processing system is shown and is generally designated 500. In a particular aspect, the processing system 500 may be incorporated into the PCD 320 described above in conjunction with FIG. 3. As shown, the processing system 500 may include a multicore central processing unit (CPU) 402 and a memory 404 connected to the multicore CPU 402. The multicore CPU 402 may include a zeroth core 410, a first core 412, and an Nth core 414.

The zeroth core 410 may include a zeroth dynamic clock and voltage scaling (DCVS) algorithm 416 executing thereon. The first core 412 may include a first DCVS algorithm 417 executing thereon. Further, the Nth core 414 may include an Nth DCVS algorithm 418 executing thereon. In a particular aspect, each DCVS algorithm 416, 417, 418 may be independently executed on a respective core 412, 414, 416.

[0019] Moreover, as illustrated, the memory 404 may include an operating system 420 stored thereon. The operating system 420 may include a scheduler 422 and the scheduler 422 may include a first run queue 424, a second run queue 426, and an Nth run queue 428. The memory 404 may also include a first application 430, a second application 432, and an Nth application 434 stored thereon.

[0020] In a particular aspect, the applications 430, 432, 434 may send one or more tasks 436 to the operating system 420 to be processed at the cores 410, 412, 414 within the multicore CPU 402. The tasks 436 may be processed, or executed, as single tasks, threads, or a combination thereof. Further, the scheduler 422 may schedule the tasks, threads, or a combination thereof for execution within the multicore CPU 402. Additionally, the scheduler 422 may place the tasks, threads, or a combination thereof in the run queues 424, 426, 428. The cores 410, 412, 414 may retrieve the tasks, threads, or a combination thereof from the run queues 424, 426, 428 as instructed, e.g., by the operating system 420 for processing, or execution, of those task and threads at the cores 410, 412, 414.

[0021] FIG. 4 also shows that the memory 404 may include a parallelism monitor 440 stored thereon. The parallelism monitor 440 may be connected to the operating system 420 and the multicore CPU 402. Specifically, the parallelism monitor 440 may be connected to the scheduler 422 within the operating system 420.

[0022] Referring to FIG. 5, a first aspect of a method of dynamically controlling the power of a central processing unit is shown and is generally designated 500. The method 500 may commence at block 502 with a do loop in which when device is powered on, the following steps may be performed.

[0023] At block 504, a power controller, e.g., a dynamic clock and voltage scaling (DCVS) algorithm, may monitor one or more CPUs. At decision 506, the power controller may determine whether the CPU is idle. If not, the method 500 may return to block 504 and continue as described herein. Otherwise, if the CPU is idle, the method 500 may proceed to block 508 and the power controller may review a busy cycle, i.e., operation window, immediately prior to the current idle state. At block 510, the power controller may determine the total work load during the previous busy cycle. Further, at block 512, the power controller may review the operational frequencies utilized during the previous busy cycle.

[0024] Moving to decision 513, the power controller may determine whether the previous busy cycle ended at the steady state level. If so, the method 500 may proceed to block 516 and the power controller may set the CPU frequency to a steady state value. Then, the method 500 may proceed to decision 518. At decision 518, the power controller may determine whether the device is powered off. If the device is powered off, the method may end. Otherwise, if the device remains powered on, the method 500 may return to block 504 and the method 500 may continue as described.

[0025] Returning to decision 513, if the previous busy cycle did not end at a the steady state, the method 500 may move to decision 514 and the power controller may determine whether the previous busy cycle included any frequency jumps or is at the maximum performance level, e.g., due to workload increases. If so, the method 500 may proceed to block 517 and the power controller may reset the longest normalized busy period. The method 500 may continue block 520 and continue as described herein.

[0026] Returning to decision 514, if the previous busy cycle did not include any frequency jumps and is not at the maximum performance level, the method 500 may proceed to block 520 and the power controller may determine the longest normalized busy period since being reset. At block 522, the power controller may determine a minimum operational frequency that would not have caused a frequency jump had it been used starting at the time the longest busy period was last reset. Next, the power controller sets the CPU frequency to the minimum frequency determined above and resets the longest busy period if the minimum frequency is not the same as the previous CPU frequency. The method 500 may then proceed to decision 518. At decision 518, the power controller may determine whether the device is powered off. If the device is powered off, the method may end. Otherwise, if the device remains powered on, the method 500 may return to block 504 and the method 500 may continue as described.

[0027] In a particular aspect, the method 500 may include a steady state and a transient state. Decision 513 may be used to control the transition between the steady state and the transient state. Having the ability to transition between the steady state and the transient state may reduce excessive oscillation in the frequency. Further, the method 500 may be considered self-tuning and may provide dynamic window sizes.

[0028] Referring to FIG. 6, a second aspect of a method of dynamically controlling the power of a central processing unit is shown and is generally designated 600. Beginning at block 602, a central processing unit (CPU) may enter an idle state. At block 604, a power controller, e.g., a dynamic clock and voltage scaling (DCVS) algorithm, may set a start idle time (StartIdleTime) equal to a current time (CurrentTime). Further, at block 606, the power controller may determine a busy time (BusyTime) by subtracting a start idle time (StartIdleTime) from an end idle time (EndIdleTime).

[0029] At block 608, the CPU may enter a software wait for interrupt (SWFI) condition. At block 610, the CPU may exit the SWFI condition. Moving to block 612, the power controller may set an end idle time (EndIdleTime) equal to a current time (CurrentTime). At decision 614, the power controller may determine whether the highest CPU frequency of

the previous busy cycle is greater than a steady state frequency. If not, the method 600 may end. Otherwise, the method 600 may proceed to decision 702 of FIG. 7.

[0030] At decision 702 of FIG. 7, the power controller may determine if the previous busy cycle included any frequency jumps or is at the maximum performance level. If so, the method may proceed to block 704 and the power controller may determine a busy time at the maximum CPU frequency (BusyTimeAtMax), i.e., how long the CPU would have been busy if it had been running at the maximum frequency during the complete busy period. The BusyTimeAtMax is a normalized value. In other words, the BusyTimeAtMax is normalized to a single performance level, e.g., the maximum frequency.

[0031] The BusyTimeAtMax may be determined using the following formula:

$$\text{BusyTimeAtMax} = \frac{\sum (\text{BusyTimeAtFreq}[N] * (N / \text{MaxCPUFreq}))}{\text{N} = \text{MinCPUFreq}}$$

where,

BusyTimeAtFreq[N] = The total amount of time that the CPU was busy during the previous busy cycle at frequency N,
MaxCPUFreq = The maximum CPU frequency; and
MinCPUFreq = The minimum CPU frequency.

[0032] For example, if the previous busy cycle was two milliseconds (2 ms) and the CPU spent one millisecond (1ms) at the a maximum frequency of one GigaHertz (1 GHz) and one millisecond at a nominal frequency of one hundred Megahertz (100 MHz), the BusyTimeAtMax would be equal to 1.1 ms.

[0033] Moving to block 706, the power controller may determine a CPU frequency (CPUFreq) that would not have caused any frequency jumps. Specifically, the power controller may determine the lowest CPU frequency that would have eliminated any frequency jumps. This is determined by calculating the slack budget for each CPU frequency that would be calculated by the transient filter and determining if the slack budget is sufficient that it would not cause the transient filter to make frequency jump.

[0034] This determination may be made by setting the CPUFreq equal to the minimum CPU frequency (MinCPUFreq) and then performing a do loop until a condition is met. Each time the condition is not met the CPUFreq may be increased by one value (CPUFreq = CPUFreq + 1) until a maximum CPU frequency (MaxCPUFreq) is met. The condition is as follows:

$$(((\text{MaxCPUFreq} * \text{slackBudget}) / (\text{MaxCPUFreq} - \text{CPUFreq})) * \text{SteadyStateAdjustment}) \leq (\text{BusyTimeAtMax} * (\text{MaxCPUFreq} / \text{CPUFreq}))$$

where,

MaxCPUFreq = The maximum CPU frequency;

slackBudget = The transient response filter deadline to jump to a higher CPU frequency for the current CPUFreq being tested;

CPUFreq = The current CPU frequency being tested to eliminate the jump;

SteadyStateAdjustment = Any adjustment required due to the steady state filter design; and

BusyTimeAtMax = The normalized value of how long the CPU was operating at the maximum CPU frequency (determined above).

[0035] Once the condition is met, the CPU frequency may be set to the CPUFreq above that meets the condition. Moving to block 708 through 712, the power controller may initialize the state of the energy minimization algorithm

described herein. Specifically, at block 708, the power controller may set a last lower time (LastLowerTime) equal to the end idle time (EndIdleTime). At block 710, the power controller may set a maximum busy time at maximum (MaxBusyTimeAtMax) equal to zero. Next, at block 712, the power controller may set a total busy time (TotalBusyTime) equal to zero. Thereafter, the method 600 may end.

[0036] Returning to decision 702, if the previous busy cycle did not include any frequency jumps and is not at the maximum performance level, the method may proceed to block 802 of FIG. 8 and the method 600 may continue as described herein.

[0037] At block 802, the power controller may determine a busy time at max (BusyTimeAtMax) using the same formula described above. Thereafter, at decision 804, the power controller may determine whether the BusyTimeAtMax is greater than a maximum busy time at the maximum CPU (MaxBusyTimeAtMax). If the BusyTimeAtMax is greater than the MaxBusyTimeAtMax, the method 600 may proceed to block 806 and the power controller may set the MaxBusyTimeAtMax equal to the BusyTimeAtMax. Then, the method 800 may move to block 808. At decision 804, if the BusyTimeAtMax is not greater than the MaxBusyTimeAtMax, the method 800 may move directly to block 808.

[0038] At block 808, the power controller may determine a running total busy time (TotalBusyTime) by adding the busy time (BusyTime) to the total busy time (TotalBusyTime). At block 810, the power controller may determine a non jumping frequency (NonJumpingFrequency) using the same do loop described above in conjunction with block 706. Moving to block 812, the power controller may determine an energy saving frequency (EnergySavingFrequency). In a particular aspect, the EnergySavingFrequency is the lowest frequency (starting from the level calculated in block 810) that the CPU should be set too in order to save on energy consumption. In this aspect, the assumption may be made that the system will not need to jump for at least as long as the amount of time since the last lowering of frequency. Also, the assumption may be made that immediately after the same time period a jump will occur. This step also includes the clock switching overhead and the scheduling overhead.

[0039] In a particular aspect, the amount of energy consumed at the current CPU frequency is determined. That value may be denoted DCVSFloorEnergy. Then, the CPU frequency may be raised until a value is found that has at most a one percent (1%) clock switch overhead, or that uses less energy than the current performance level. In a particular aspect, the one percent (1%) value is arbitrary and may be eliminated.

[0040] While the CPU frequency is less than the current CPU frequency and the elapsed time (elapsedTime) is less than clockSwitchOverhead times 2 times one hundred, the system may determine how long to run at the CPU frequency in order to see the exact same workload as the previous busy cycle. That value may be denoted as the DCVSJFloorBusyTime and may be determined using the following formula:

$$\text{DCVSJFloorBusyTime} = (\text{totalBusyTime} * \text{CPUFreq}/\text{currentCPUFreq}) + (\text{clockSwitchOverhead} * 2)$$

[0041] Also, the system may determine the amount of energy that the CPU would consume at the CPUFreq with the same workload as the previous busy cycle. Moving to decision 814, the power controller may determine whether the EnergySavingFrequency is less than the current CPUFrequency. If so, the method 600 may return to block 708 of FIG. 7 and the method 600 may continue as described herein. Otherwise, the method 600 may end.

[0042] It is to be understood that the method steps described herein need not necessarily be performed in the order as described. Further, words such as "thereafter," "then," "next," etc. are not intended to limit the order of the steps. These words are simply used to guide the reader through the description of the method steps. Moreover, the methods described herein are described as executable on a portable computing device (PCD). The PCD may be a mobile telephone device, a portable digital assistant device, a smartbook computing device, a netbook computing device, a laptop computing device, a desktop computing device, or a combination thereof.

[0043] In a particular aspect, a DCVS algorithm is a mechanism which measures CPU load/idle time and dynamically adjusts the CPU clock frequency to track the workload in an effort to reduce power consumption while still providing satisfactory system performance. As the workload changes, the change in CPU throughput may track, but also necessarily lag, the changes in the workload. Unfortunately, this may introduce a problem in cases where the workload has Quality of Service (QoS) requirements, as the DCVS algorithm may not track the workload quickly enough. Further, tasks may fail. The performance (QoS) issues may be solved with the introduction of transient performance deadlines, i.e., explicit panics to a higher performance level, however this may result in an actual increase in power due frequency oscillations induced when transitioning between steady state and transient CPU frequencies.

[0044] The system and methods described herein may be used to manage the transition between transient and steady state performance levels. Further, the system and methods described herein may substantially reduce any oscillation. As a result, there may be substantial savings in net power consumed. As shown in FIG. 9, the systems and methods described herein provide dynamic CPU power control without excessive oscillation.

[0045] In order to avoid the problem of excessive frequency oscillations due to QoS deadlines and/or explicit panics, the present methods introduce an energy minimization algorithm which may control the transitions between the steady state and transient state, i.e., explicit panics to a higher performance levels. The energy minimization methods, or algorithms, described herein may effectively managing the jumps between the maximum performance level caused by the transient response guarantee, i.e., the explicit jumps to higher performance levels, and the lower steady state performance level.

[0046] In the absence of any jumps between the two levels, the energy minimization algorithm can just set the CPU performance level to the steady state value. As such, the methods described herein may determine how to lower the performance level down to the steady state level in the most energy efficient manner. Further, these methods may actively manage the performance level from the moment in time that a transient pulse, i.e., an explicit panic to a higher frequency, completes, until the performance level is taken back down to the level indicated by the steady state level. In general, the performance level may be taken down in discrete steps that will eliminate the possibility of needing a jump to a higher performance level if the exact same idle/busy profile was repeated, that was just seen since the last drop in frequency (performance level). For example, if a transient pulse causes a jump to the maximum clock frequency, then on the next idle period, the energy minimization methods, described herein, may set the performance level to that which would have eliminated the jump. On each succeeding idle period a controller may determine the lowest frequency at or above the steady state performance level that would have saved energy, assuming that the exact same idle/busy profile is repeated from the point in time that the last frequency reduction was made.

[0047] In a particular aspect, the methods described herein may utilize other approaches to reduce the frequency from the higher performance level down to the steady state level. For example, steps may be time, steps may be linear, steps may be non-linear, a low pass filter based on jumps per second may be used, or any combination thereof may be used.

[0048] Further, the system and method described herein may ensure that excessive frequency changes may not be made, despite the presence of QoS deadlines, or explicit panics to higher performance levels. Accordingly, power consumption may be substantially lowered.

[0049] In one or more exemplary aspects, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer program product such as a machine readable medium, i.e., a computer-readable medium. Computer-readable media includes both computer storage media and communication media including any medium that facilitates transfer of a computer program from one place to another. A storage media may be any available media that may be accessed by a computer. By way of example, and not limitation, such computer-readable media may comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that may be used to carry or store desired program code in the form of instructions or data structures and that may be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if the software is transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and blu-ray disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

Claims

1. A method of dynamically controlling power within a central processing unit, the method comprising:
 - entering an idle state;
 - during the idle state, reviewing a previous busy cycle immediately prior to the idle state; and
 - based on the review of the previous busy cycle, determining a CPU frequency for a next busy cycle.
2. The method of claim 1, further comprising:
 - determining a total work load during the previous busy cycle.
3. The method of claim 2, further comprising:
 - reviewing operational frequencies during the previous busy cycle.

4. The method of claim 3, further comprising:

determining whether the previous busy cycle ended at a steady state level.

5 5. The method of claim 4, further comprising:

setting a CPU frequency for the next busy cycle to the steady state value when the previous busy cycle ended at the steady state level.

10 6. The method of claim 4, further comprising:

determining whether any frequency jumps occurred during the busy cycle or whether the previous busy cycle ended at a maximum performance level when the previous busy cycle did not end at the steady state level.

15 7. The method of claim 6, further comprising:

resetting a longest busy period to zero when any frequency jumps occurred or when the previous cycle ended at the steady state level.

20 8. The method of claim 7, further comprising:

determining a longest normalized busy period since the last reset.

25 9. The method of claim 8, further comprising:

determining a minimum frequency that would have eliminated any frequency jumps during the previous cycle.

10. The method of claim 9, further comprising:

30 setting a CPU frequency for the next busy cycle equal to the minimum frequency that would have eliminated any frequency jumps.

11. A device, comprising:

35 means for implementing the method according to any of Claims 1 to 10.

12. The device according to Claim 11 wherein the means for implementing comprises a processor.

40 13. A computer readable storage medium carrying a computer program stored thereon, said program comprising computer executable instructions adapted to perform the method steps of any of Claims 1 to 10 when executed by a processing module.

45 Patentansprüche

1. Ein Verfahren zum dynamischen Steuern von Leistung in einer zentralen Verarbeitungseinheit, wobei das Verfahren Folgendes aufweist:

50 Eintreten in einen Leerlaufzustand;
während des Leerlaufzustandes, Überprüfen eines vorherigen beschäftigten Zyklus unmittelbar vor dem Leerlaufzustand; und
basierend auf der Überprüfung des vorherigen beschäftigten Zyklus, Bestimmen einer CPU-Frequenz für einen nächsten beschäftigten Zyklus.

55 2. Verfahren nach Anspruch 1, das weiter Folgendes aufweist:

Bestimmen einer Gesamtarbeitslast während des vorherigen beschäftigten Zyklus.

3. Verfahren nach Anspruch 2, das weiter Folgendes aufweist:

Überprüfen von Betriebsfrequenzen während des vorherigen beschäftigten Zyklus.

- 5 4. Verfahren nach Anspruch 3, das weiter Folgendes aufweist:

Bestimmen, ob der vorherige beschäftigte Zyklus zuvor beschäftigt auf einem Gleichgewichtsniveau geendet hat.

- 10 5. Verfahren nach Anspruch 4, das weiter Folgendes aufweist:

Einstellen einer CPU-Frequenz für den nächsten beschäftigten Zyklus auf den Gleichgewichtswert, wenn der vorherige beschäftigte Zyklus auf einem Gleichgewichtsniveau geendet hat.

- 15 6. Verfahren nach Anspruch 4, das weiter Folgendes aufweist:

Bestimmen, ob irgendwelche Frequenzsprünge während des beschäftigten Zyklus aufgetreten sind oder ob der vorherige beschäftigte Zyklus auf einem maximalen Performance- bzw. Leistungsniveau geendet hat, wenn der vorherige beschäftigte Zyklus nicht auf dem Gleichgewichtsniveau geendet hat.

- 20 7. Verfahren nach Anspruch 6, das weiter Folgendes aufweist:

Zurücksetzen einer längsten beschäftigten Periode auf Null, wenn irgendwelche Frequenzsprünge aufgetreten sind oder wenn der vorherige Zyklus auf dem Gleichgewichtsniveau geendet hat.

- 25 8. Verfahren nach Anspruch 7, das weiter Folgendes aufweist:

Bestimmen einer längsten normierten beschäftigten Periode seit der letzten Zurücksetzung.

- 30 9. Verfahren nach Anspruch 8, das weiter Folgendes aufweist:

Bestimmen einer minimalen Frequenz, die jegliche Frequenzsprünge während des vorherigen Zyklus eliminiert haben würde.

- 35 10. Verfahren nach Anspruch 9, das weiter Folgendes aufweist:

Einstellen einer CPU-Frequenz für den nächsten beschäftigten Zyklus auf einen Wert gleich der minimalen Frequenz, die jegliche Frequenzsprünge eliminiert haben würde.

- 40 11. Eine Einrichtung, die Folgendes aufweist:

Mittel zum Implementieren des Verfahrens gemäß einem der Ansprüche 1 bis 10.

- 45 12. Einrichtung nach Anspruch 11, wobei die Mittel zum Implementieren einen Prozessor aufweisen.

- 50 13. Ein computerlesbares Speichermedium, das ein darauf gespeichertes Computerprogramm trägt, wobei das Programm von einem Computer ausführbare Instruktionen aufweist, die ausgelegt sind zum Durchführen der Verfahrensschritte nach einem der Ansprüche 1 bis 10, wenn sie durch ein Verarbeitungsmodul ausgeführt werden.

Revendications

- 55 1. Procédé de contrôle dynamique de la puissance dans une unité centrale de traitement, le procédé comprenant les étapes consistant à :

passer dans un état de repos ;
pendant l'état de repos, examiner un cycle d'activité précédent, immédiatement avant l'état de repos ; et
en fonction de l'examen du cycle d'activité précédent, l'étape consistant à déterminer une fréquence de CPU

pour un cycle d'activité suivant.

- 2.** Procédé selon la revendication 1, comprenant en outre l'étape consistant à :

déterminer une charge de travail totale pendant le cycle d'activité précédent.

- 3.** Procédé selon la revendication 2, comprenant en outre l'étape consistant à :

examiner les fréquences opérationnelles pendant le cycle d'activité précédent.

- 4.** Procédé selon la revendication 3, comprenant en outre l'étape consistant à :

déterminer si le précédent cycle d'activité s'est terminé à un niveau d'un état stationnaire.

- 5.** Procédé selon la revendication 4, comprenant en outre l'étape consistant à :

définir une fréquence de CPU pour le prochain cycle d'activité à la valeur d'un état stationnaire lorsque le cycle d'activité précédent s'est terminé au niveau d'un état stationnaire.

- 6.** Procédé selon la revendication 4, comprenant en outre l'étape consistant à :

déterminer si d'éventuels sauts de fréquence sont survenus pendant le cycle d'activité ou si le cycle d'activité précédent s'est terminé à un niveau de performance maximum lorsque le cycle d'activité précédent ne s'est pas terminé au niveau d'un état stationnaire.

- 7.** Procédé selon la revendication 6, comprenant en outre l'étape consistant à :

remettre à zéro une période d'activité la plus longue lorsque d'éventuels sauts de fréquence se sont produits ou lorsque le cycle précédent s'est terminé au niveau d'un état stationnaire.

- 8.** Procédé selon la revendication 7, comprenant en outre l'étape consistant à :

déterminer une période d'activité normalisée la plus longue depuis la dernière remise à zéro.

- 9.** Procédé selon la revendication 8, comprenant en outre l'étape consistant à :

déterminer une fréquence minimale qui aurait éliminé d'éventuels sauts de fréquence pendant le cycle précédent.

- 10.** Procédé selon la revendication 9, comprenant en outre l'étape consistant à :

définir une fréquence de CPU pour le prochain cycle d'activité égale à la fréquence minimale qui aurait éliminé d'éventuels sauts de fréquence.

- 11.** Dispositif comprenant :

des moyens pour mettre en oeuvre le procédé selon l'une quelconque des revendications 1 à 10.

- 12.** Le procédé selon la revendication 11, dans lequel les moyens pour mettre en oeuvre comprennent un processeur.

- 13.** Un support de stockage lisible par ordinateur portant un programme informatique stocké sur celui-ci, ledit programme comprenant des instructions exécutables par ordinateur adaptées pour mettre en oeuvre les étapes du procédé selon l'une quelconque des revendications 1 à 10 lors de leur exécution par un module de traitement.

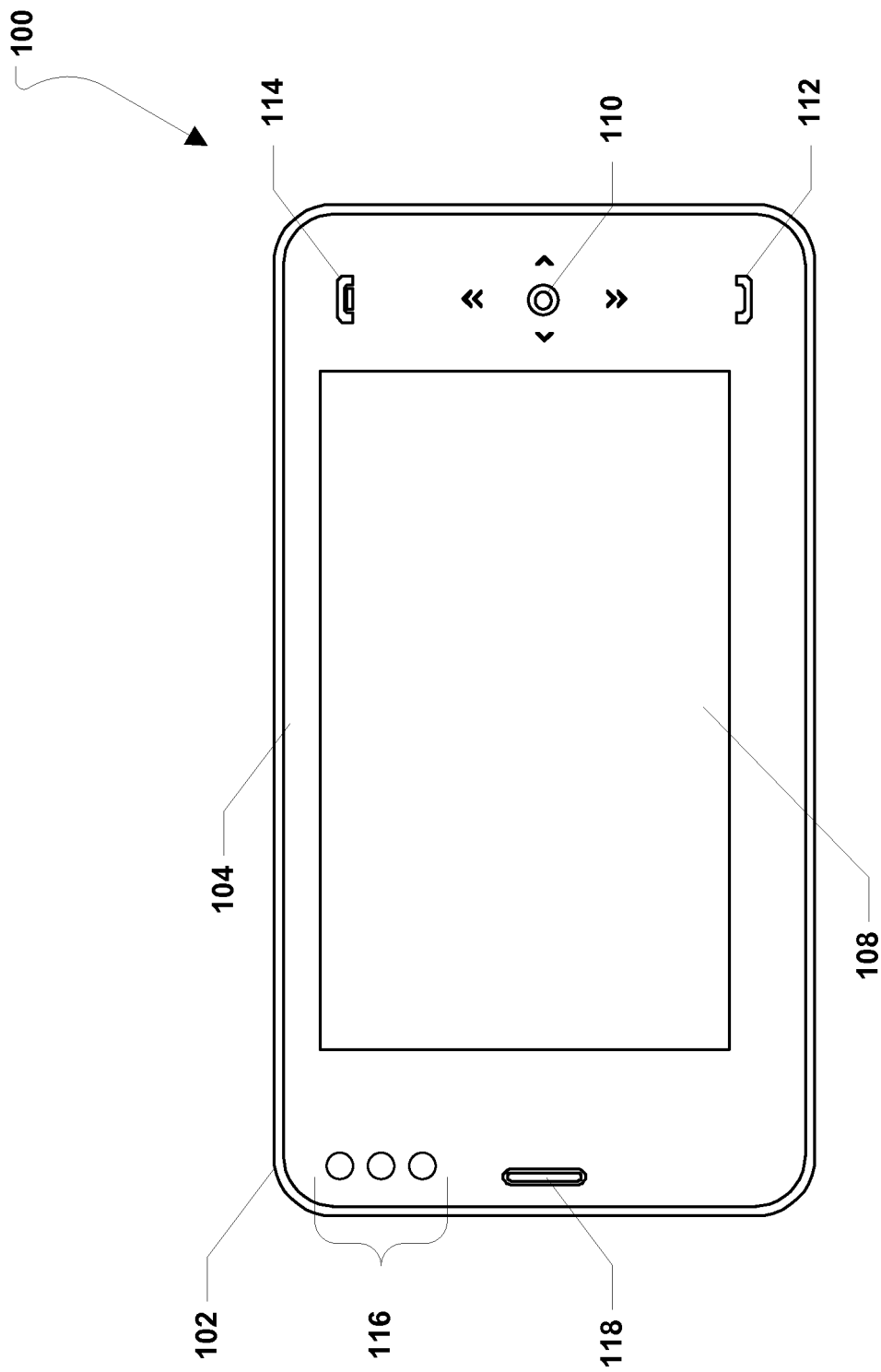


FIG. 1

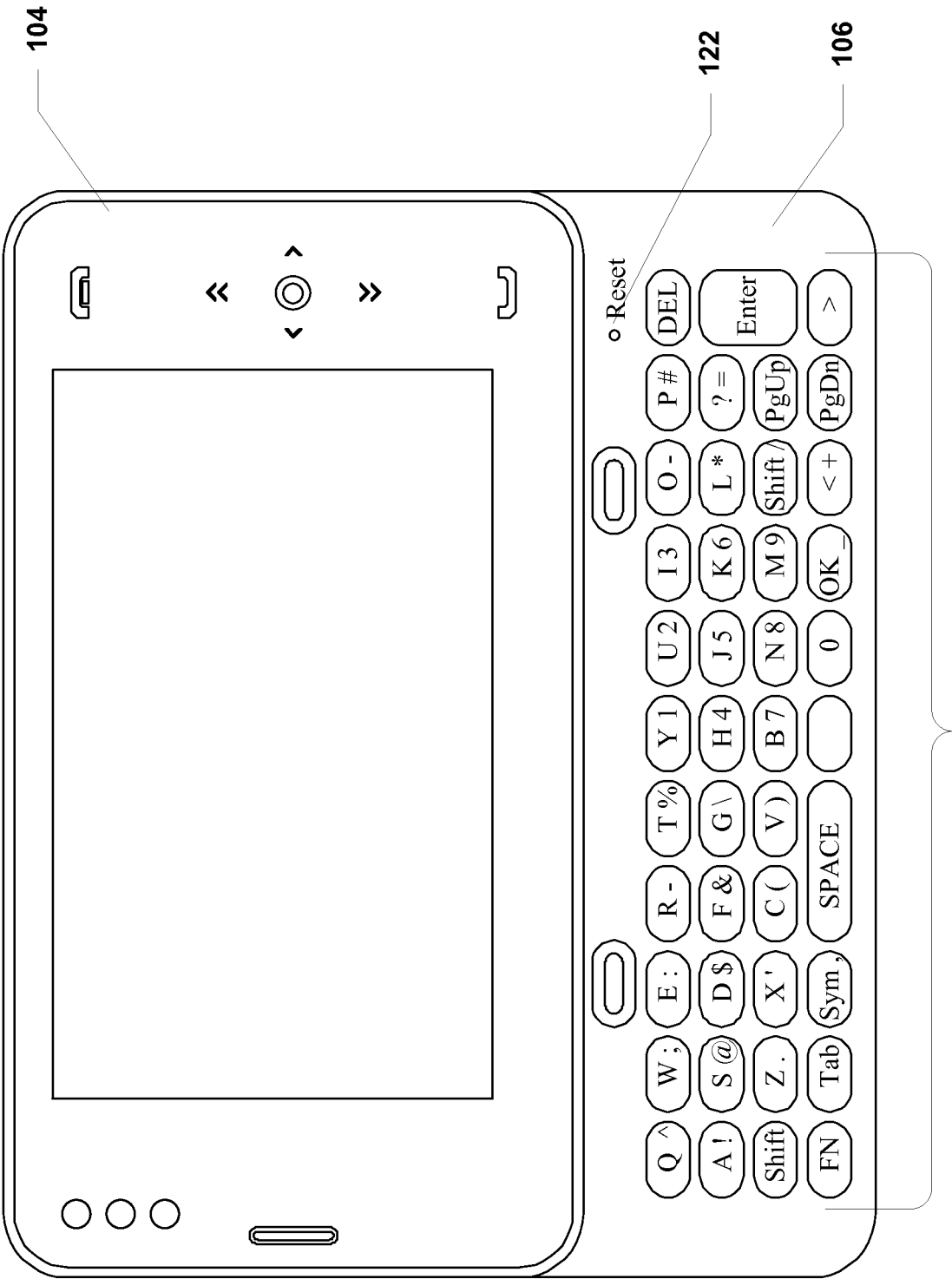


FIG. 2

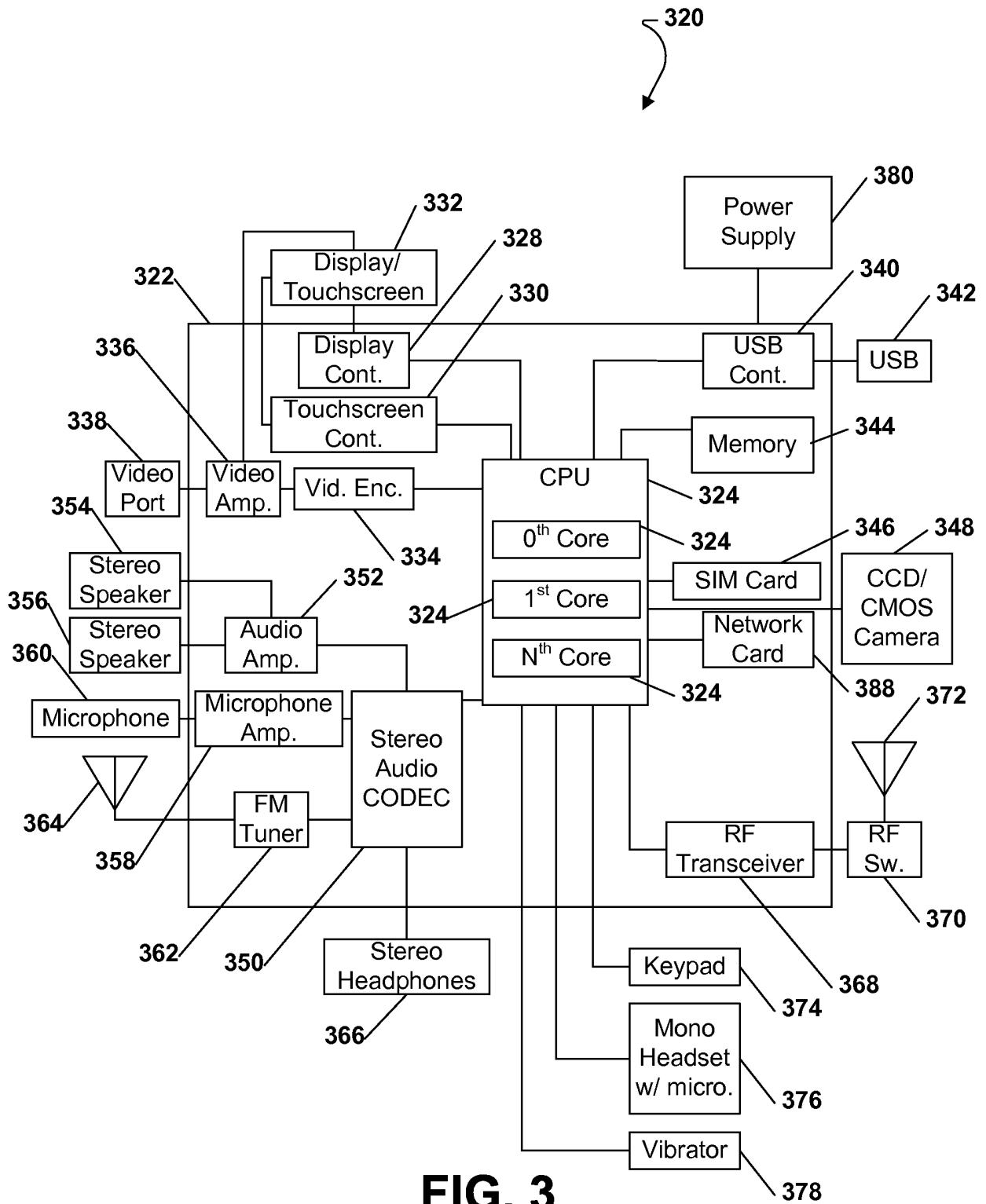


FIG. 3

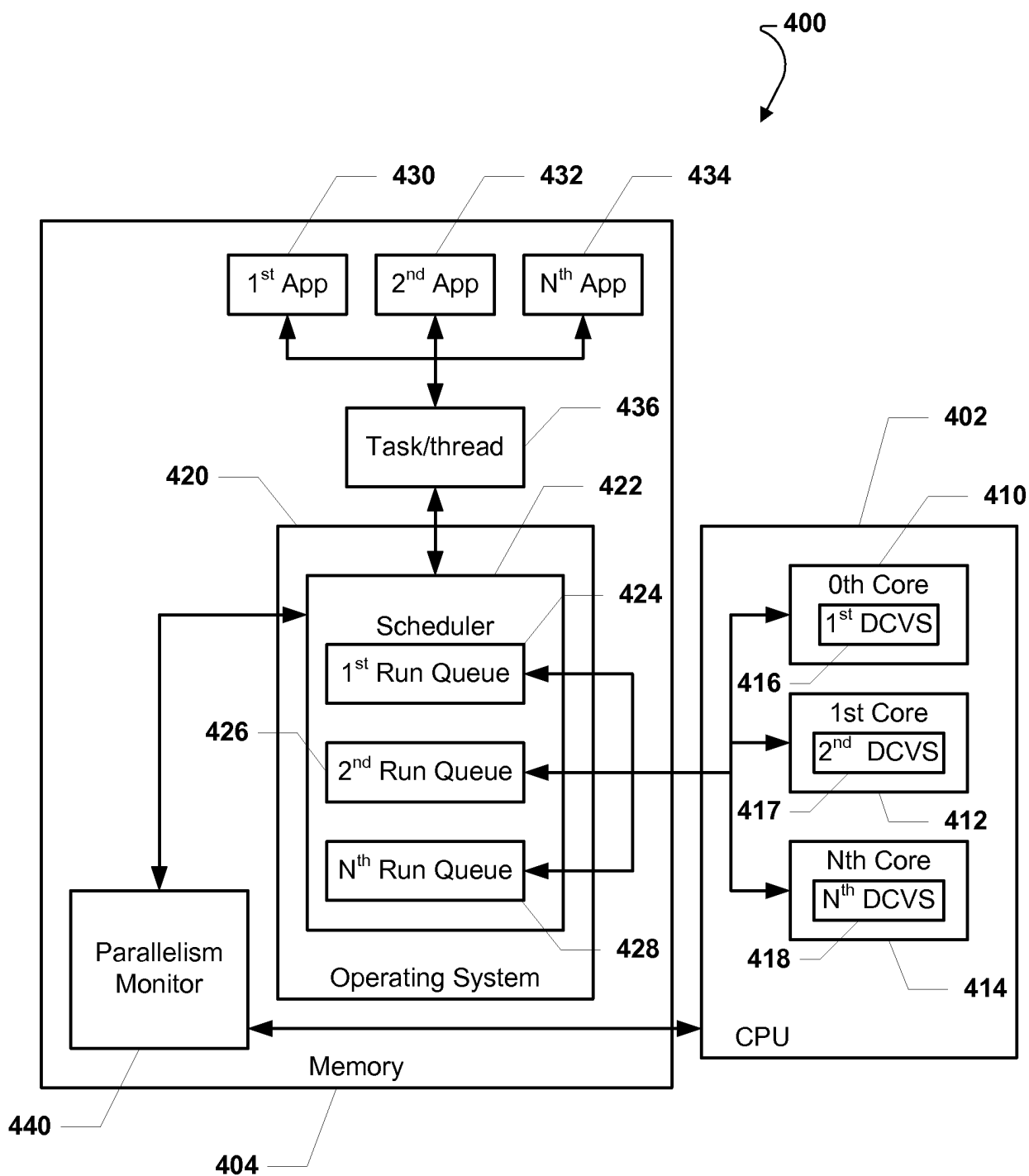
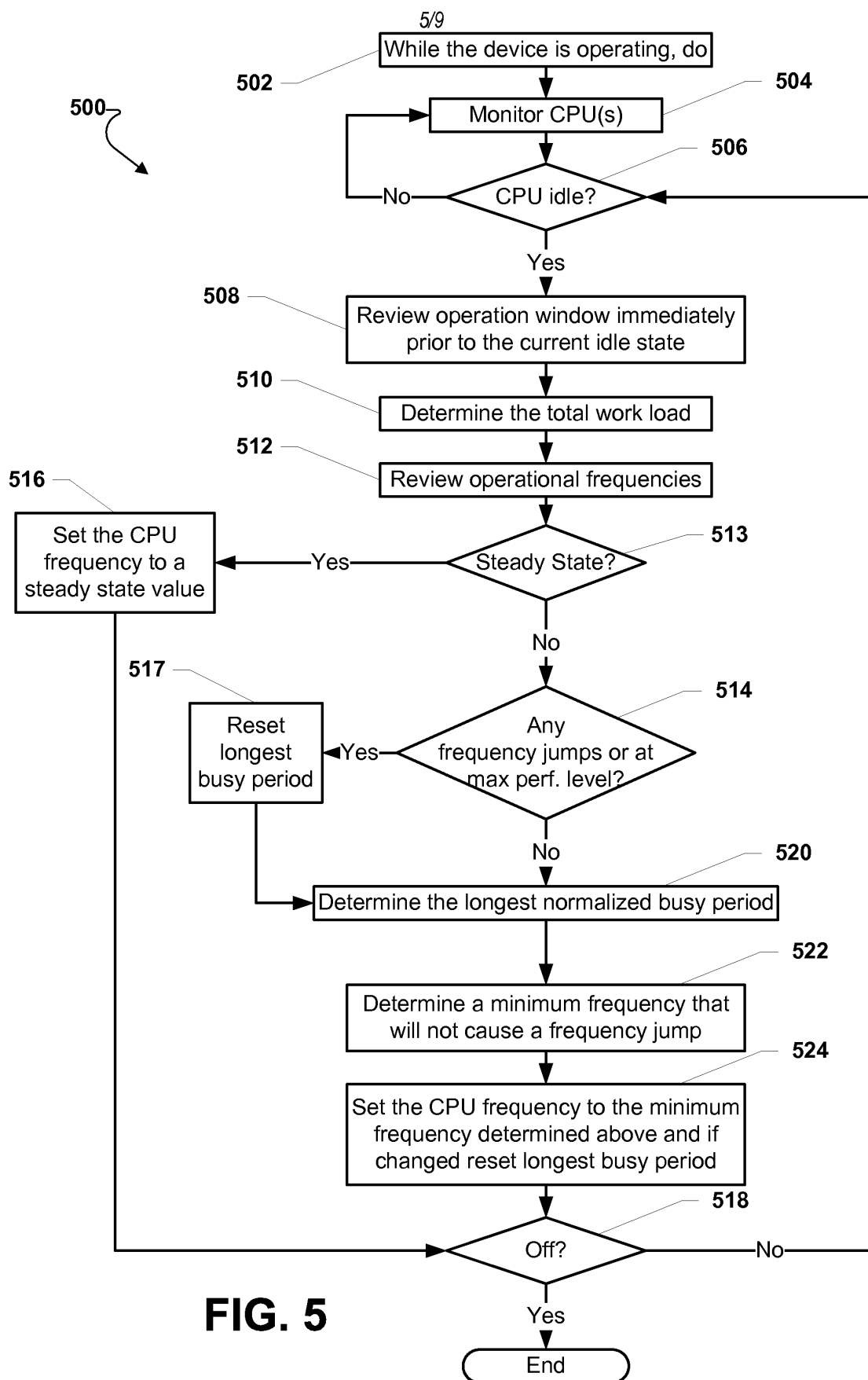
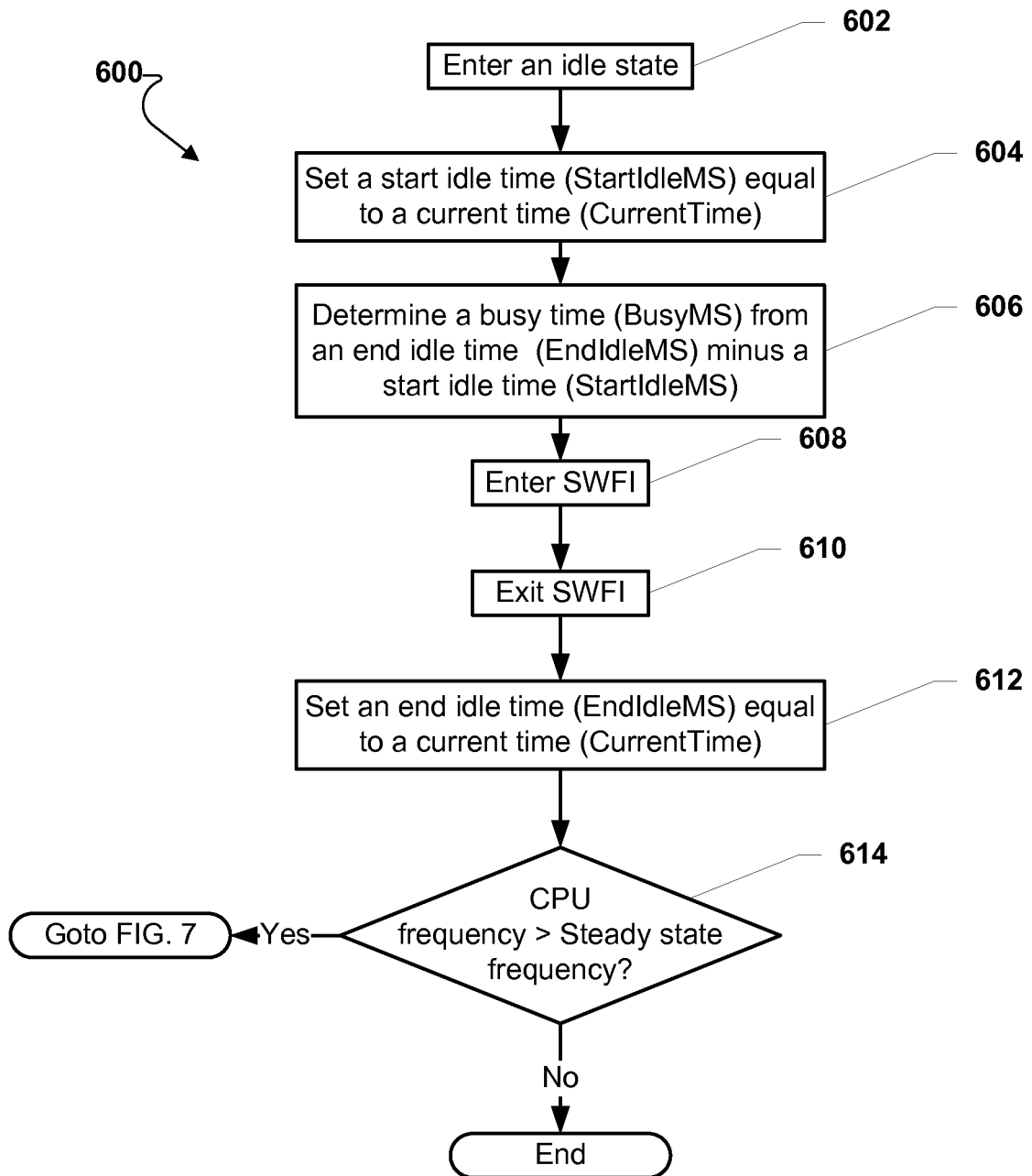
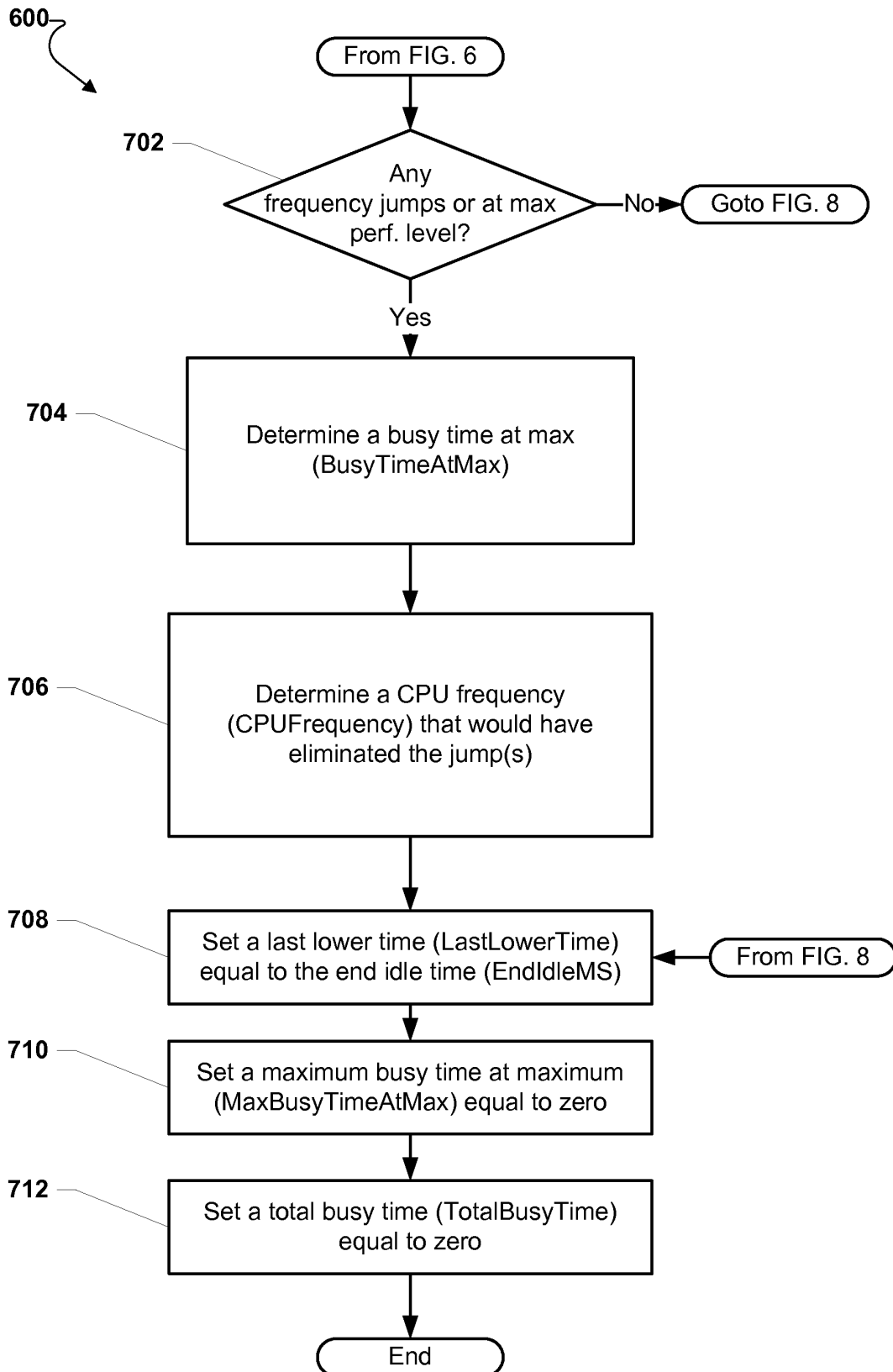
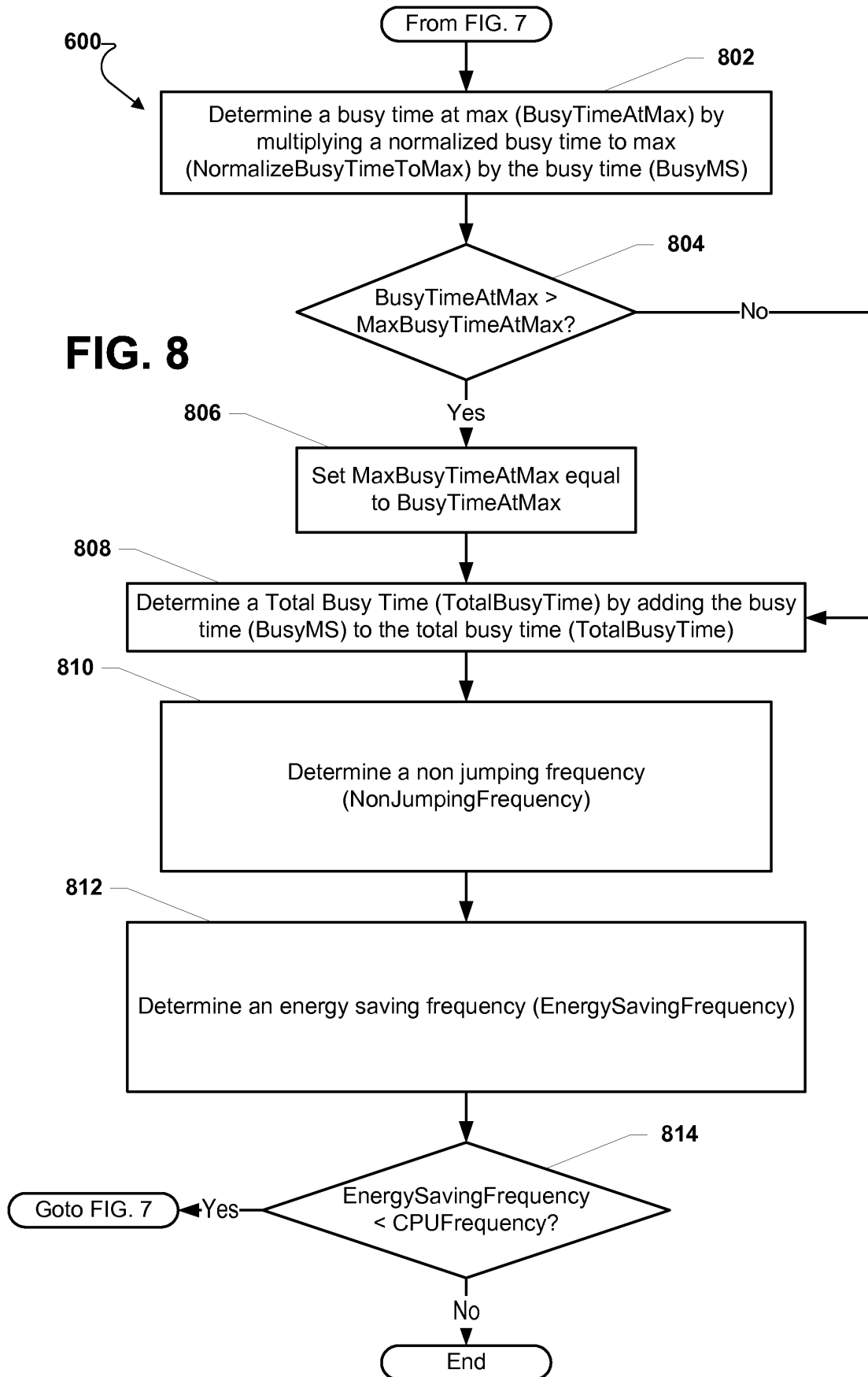


FIG. 4



**FIG. 6**

**FIG. 7**



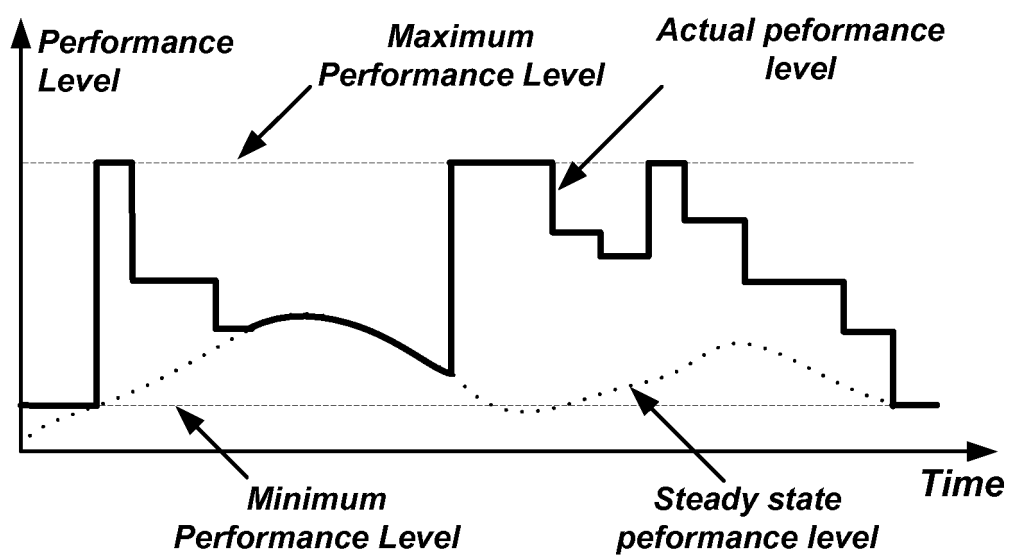


FIG. 9

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 20060123253 A [0002]